

[illegible]

Group Art Unit: 2122

Examiner: Chavis, J.

Atty. Docket: 40.0010 C1

Date of Deposit October 23, 2001
I hereby certify under 37 CFR 1.10 that this correspondence is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" with sufficient postage on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

Mary L. Thompson

Mary L. Thompson

1

expensive and data in RAM is lost when power is removed. A microprocessor system typically has relatively little ROM and EEPROM, and has 1 to 128 megabytes of RAM, since it is not constrained by what will fit on a single integrated circuit device, and often has access to an external disk memory system that serves as a large writable, non-volatile storage area at a lower cost than EEPROM. However, a microcontroller typically has a small RAM of 0.1 to 2.0 K, 2K to 8K of EEPROM, and 8K – 56K of ROM.--

Please replace the paragraph beginning at page 5, line 3, with the following rewritten paragraph:

--Due to the concern for security, applications written for integrated circuit cards have unique properties. For example, each application typically is identified with a particular owner or identity. Because applications typically are written in a low-level programming language, such as assembly language, the applications are written for a particular type of microcontroller. Due to the nature of low level programming languages, unauthorized applications may access data on the integrated circuit card. Programs written for an integrated circuit card are identified with a particular identity so that if two identities want to perform the same programming function there must be two copies of some portions of the application on the microcontroller of the integrated circuit card.--

Please replace the paragraph beginning at page 6, line 25, with the following rewritten paragraph:

--Among the advantages of the invention are one or more of the following. New applications may be downloaded to a smart card without compromising the security of the smart card. These applications may be provided by different companies loaded at different times using different terminals. Security is not compromised since the applications are protected against unauthorized access of any application code or data by the security features provided by the Java virtual machine. Smart card applications can be created in high level languages such as Java and Eiffel, using powerful mainstream program development tools. New applications can be quickly prototyped and downloaded

to a smart card in a matter of hours without resorting to soft masks. Embedded systems using microcontrollers can also gain many of these advantages for downloading new applications, high level program development, and rapid prototyping by making use of this invention.--

Please replace the paragraph beginning at page 15, line 24, with the following rewritten paragraph:

--The terminal 14 can also interact with applications running in the integrated circuit card 10. In some cases, different terminals may be used for these purposes. For example, one kind of terminal may be used to prepare applications, different terminals could be used to download the applications, and yet other terminals could be used to run the various applications. Terminals can be automated teller machines (ATMs), point-of-sale terminals, door security systems, toll payment systems, access control systems, or any other system that communicates with an integrated circuit card or microcontroller.--

Please replace the paragraph beginning at page 18, line 6, with the following rewritten paragraph:

--To avoid dynamic linking in the card, all the information that is distributed across several Java class files 24a, 24b, and 24c that form the application 24, are coalesced into one card class file 27 by the process shown in the flowchart in Fig. 5. The first class file to be processed is selected 51a. The constant pool 42 is compacted 51b in the following manner. All objects, classes, fields, methods referenced in a Java class file 24a are identified by using strings in the constant pool 42 of the class file 24a. The card class file converter 26 compacts the constant pool 42 found in the Java class file 24a into an optimized version. This compaction is achieved by mapping all the strings found in the class file constant pool 42 into integers (the size of which is microcontroller architecture dependent). These integers are also referred to as IDs. Each ID uniquely identifies a particular object, class, field or method in the application 20. Therefore, the card class file converter 26 replaces the strings in the Java class file constant pool 42 with

its corresponding unique ID. Appendix B shows an example application HelloSmartCard.java, with a table below illustrating the IDs corresponding to the strings found in the constant pool of the class file for this application. The IDs used for this example are 16-bit unsigned integers.--

In the Drawings:

A Proposed Drawing Amendment for Approval by the Examiner accompanies this communication. The proposed corrections to Figs. 4, 16 & 18 are marked in red ink on copies of the original drawings.

In the Claims:

Please cancel Claims 1-105 without prejudice.

Please add the following new Claims:

145. A microcontroller comprising:

a memory storing:

a derivative application derived from an application having a class file format wherein the application is derived from an application having a class file format by first compiling the application having a class file format into a compiled form and then converting the compiled form into a converted form, and

an interpreter configured to interpret applications derived from applications having a class file format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the derivative application for execution.

146. The microcontroller of claim 145, further comprising:

a communicator configured to communicate with a terminal.

147. The microcontroller of claim 147, wherein the terminal has a card reader and the communicator comprises a contact for communicating with the card reader.

148. The microcontroller of claim 147, wherein the terminal has a wireless communicator and a wireless transceiver for communicating with the wireless communication device.

149. The microcontroller of claim 147, wherein the terminal has a wireless communication device and the communicator comprises a wireless transmitter for communicating with the wireless communication device.

150. The microcontroller of claim 145, wherein the class file format comprises a Java class file format.

151. A microcontroller having a set of resource constraints and comprising:
a memory, and
an interpreter loaded in memory and operable within the set of resource constraints,
the microcontroller having: at least one application loaded in the memory to be interpreted by the interpreter, wherein the at least one application is generated by a programming environment comprising:
a) a compiler for compiling application source programs written in high level language source code form into a compiled form, and
b) a converter for post processing the compiled form into a minimized form suitable for interpretation within the set of resource constraints by the interpreter.

152. The microcontroller of Claim 151, wherein the compiled form includes attributes, and the converter comprises a means for including attributes required by the interpreter while not including the attributes not required by the interpreter.

153. The microcontroller of Claim 151 wherein the compiled form is in a standard Java class file format and the converter accepts as input the compiled form in the standard Java class file format and produces output in a form suitable for interpretation by the interpreter.

154. The microcontroller of Claim 151 wherein the compiled form includes associating an identifying string for objects, classes, fields, or methods, and the converter comprises a means for mapping such strings to unique identifiers.

155. The microcontroller of Claim 154 wherein each unique identifier is an integer.

156. The microcontroller of Claim 154 wherein the mapping of strings to unique identifiers is stored in a string to identifier map file.

157. The microcontroller of Claim 151 where in the high level language supports a first set of features and a first set of data types and the interpreter supports a subset of the first set of features and a subset of the first set of data types, and wherein the converter verifies that the compiled form only contains features in the subset of the first set of features and only contains data types in the subset of the first set of data types.

158. The microcontroller of Claim 154 wherein the compiled form is in a byte code format and the converter comprises means for translating from the byte codes in the compiled form to byte codes in a format suitable for interpretation by the interpreter by:

using at least one step in a process including the steps:

- a) recording all jumps and their destinations in the original byte codes;
- b) converting specific byte codes into equivalent generic byte codes or vice-versa;
- c) modifying byte code operands from references using identifying strings to references using unique identifiers; and
- d) renumbering byte codes in the compiled form to equivalent byte codes in the format suitable for interpretation; and

relinking jumps for which destination address is effected by conversion step a), b), c), or d).

159. The microcontroller of Claim 151 wherein the application program is compiled into a compiled form for which resources required to execute or interpret the compiled form exceed those available on the microcontroller.

160. The microcontroller of Claim 151 wherein the compiled form is designed for portability on different computer platforms.

161. The microcontroller of Claim 151 wherein the interpreter is further configured to determine, during an interpretation of an application, whether the application meets a security criteria selected from a set of rules containing at least one rule selected from the set:

- not allowing the application access to unauthorized portions of memory,
- not allowing the application access to unauthorized microcontroller resources,
- wherein the application is composed of byte codes and checking a plurality of byte codes at least once prior to execution to verify that execution of the byte codes does not violate a security constraint.

162. The microcontroller of Claim 151 wherein at least one application program is generated by a process including the steps of:

- prior to loading the application verifying that the application does not violate any security constraints; and
- loading the application in a secure manner.

163. The microcontroller of Claim 162 wherein the step of loading in a secure manner comprises the step of:

- verifying that the loading identity has permission to load applications onto the microcontroller.

164. The microcontroller of Claim 162 wherein the step of loading in a secure manner comprises the step of:

encrypting the application to be loaded using a loading key.

165. A method of programming a microcontroller having a memory and a processor operating according to a set of resource constraints, the method comprising the steps of:

inputting an application program in a first programming language;

compiling the application program in the first programming language into a first intermediate code associated with the first programming language, wherein the first intermediate code being interpretable by at least one first intermediate code virtual machine;

converting the first intermediate code into a second intermediate code; wherein the second intermediate code is interpretable within the set of resource constraints by at least one second intermediate code virtual machine; and

loading the second intermediate code into the memory of the microcontroller.

166. The method of programming a microcontroller of Claim 165 wherein the step of converting further comprises:

associating an identifying string for objects, classes, fields, or methods; and

mapping such strings to unique identifiers.

167. The method of Claim 166 wherein the step of mapping comprises the step of mapping strings to integers.

168. The method of Claim 165 wherein the step of converting comprises at least one of the steps of:

a) recording all jumps and their destinations in the original byte codes;

b) converting specific byte codes into equivalent generic byte codes or vice-versa;

c) modifying byte code operands from references using identifying strings to references using unique identifiers;

- d) renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation; and
- e) relinking jumps for which destination address is effected by conversion step a), b), c), or d).

169. The method of Claim 165 wherein the step of loading the second intermediate code into the memory of the microcontroller further comprises checking the second intermediate code prior to loading the second intermediate code to verify that the second intermediate code meets a predefined integrity check and that loading is performed according to a security protocol.

170. The method of Claim 169 wherein the security protocol requires that a particular identity must be validated to permit loading prior to the loading of the second intermediate code.

171. The method of Claim 169 further characterized by providing a decryption key and wherein the security protocol requires that the second intermediate code is encrypted using a loading key corresponding to the decryption key.

172. A microcontroller operable to execute derivative programs which are derivatives of programs written in an interpretable programming language having a memory and an interpreter, the microcontroller comprising:

- (a) the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the interpretable programming language; and
- (b) the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the interpretable language wherein a derivative of a program written in the interpretable programming language is derived from the program written in the interpretable programming language by applying at least one rule selected from a set of rules including:
 - (1) mapping strings to identifiers;

- (2) performing security checks prior to or during interpretation;
- (3) performing structural checks prior to or during interpretation; and
- (4) performing semantic checks prior to or during interpretation.

173. The microcontroller of Claim 172 wherein the derivative programs are class files or derivatives of class files.

174. The microcontroller of Claim 172 further comprising:
the memory containing less than 1 megabyte of storage.

175. The microcontroller of Claim 172 wherein the security checks the microcontroller is further comprising:

- (c) logic to receive a request from a requester to access one of a plurality of derivative programs;
- (d) after receipt of the request, determine whether the one of a plurality of derivative programs complies with a predetermined set of rules; and
- (e) based on the determination, selectively grant access to the requester to the one of the plurality of applications.

176. The microcontroller of Claim 175, wherein the predetermined rules are enforced by the interpreter while the derivative program is being interpreted by determining whether the derivative program has access rights to a particular part of memory the derivative program is attempting to access.

177. The microcontroller of Claim 172 further wherein the microcontroller is configured to perform at least one security check selected from the set having the members:

- (a) enforcing predetermined security rules while the derivative program is being interpreted, thereby preventing the derivative program from accessing

unauthorized portions of memory or other unauthorized microcontroller resources,

- (b) the interpreter being configured to check each bytecode at least once prior to execution to determine that the bytecode can be executed in accordance with pre-execution and post-execution checks, and
- (c) the derivative program is checked prior to being loaded into the microcontroller to verify the integrity of the derivative program and loading is performed according to a security protocol.

178. The microcontroller of Claim 177 wherein the security protocol requires that a particular identity must be validated to permit loading a derivative program onto a card.

179. The microcontroller of Claim 177 further comprising a decryption key wherein the security protocol requires that a derivative program to be loaded is encrypted using a loading key corresponding to the decryption key.

180. The microcontroller of Claim 172 wherein the microcontroller is configured to provide cryptographic services selected from the set including encryption, decryption, signing, signature verification, mutual authentication, transport keys, and session keys.

181. The microcontroller of Claim 172 further comprising a file system and wherein the microcontroller is configured to provide secure access to the file system through a means selected from the set including:

- (a) the microcontroller having access control lists for authorizing reading from a file, writing to a file, or deletion of a file,
- (b) the microcontroller enforcing key validation to establish the authorized access to a file, and
- (c) the microcontroller verifying card holder identity to establish the authorized access to a file.

182. An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

an application derived from a program written in a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

an interpreter operable to interpret such an application derived from a program written in a high level programming language format; and

a processor coupled to the memory, the processor configured to use the interpreter to interpret the application for execution and to use the communicator to communicate with the terminal.

183. The integrated circuit card of Claim 182 wherein the converting step further comprises:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

184. A method for use with an integrated circuit card and a terminal, comprising:

storing an interpreter operable to interpret programs derived from programs written in a high level programming language and an application derived from a program written in a high level programming language format in a memory of the integrated circuit card wherein the application is derived from a program written in a high level programming language format by first compiling

the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

using a processor of the integrated circuit card to use the interpreter to interpret the application for execution; and

using a communicator of the card when communicating between the processor and the terminal.

185. The method of Claim 184 wherein the converting step further comprises:

recording all jumps and their destinations in the original byte codes;

converting specific byte codes into equivalent generic byte codes or vice-versa; and

renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

186. An integrated circuit card for use with a terminal, comprising:

a communicator configured to communicate with the terminal;

a memory storing:

applications, each application derived from applications having a high level programming language format, and

an interpreter operable to interpret applications derived from applications having a high level programming language format wherein the application is derived from a program written in a high level programming language format by first compiling the program into a compiled form and then converting the compiled form into a converted form, the converting step including modifying byte code operands from references using identifying strings to references using unique identifiers; and

a processor coupled to the memory, the processor configured to:

- a.) use the interpreter to interpret the applications for execution,
- b.) use the interpreter to create a firewall to isolate the applications from each other, and
- c.) use the communicator to communicate with the terminal.

187. The integrated circuit card of Claim 186 wherein the interpreter is further operable to interpret applications derived using a converting step including:

- recording all jumps and their destinations in the original byte codes;
- converting specific byte codes into equivalent generic byte codes or vice-versa; and
- renumbering byte codes in a compiled format to equivalent byte codes in a format suitable for interpretation.

188. A microcontroller operable to execute derivative programs which are derivatives of programs written in an interpretable programming language having a memory and an interpreter, the microcontroller comprising:

- the microcontroller operating within a set of resource constraints including the memory being of insufficient size to permit interpretation of programs written in the interpretable programming language; and
- the memory containing an interpreter operable to interpret the derivative programs written in the derivative of the interpretable language wherein a derivative of a program written in the interpretable programming language is derived from the program written in the interpretable programming language by mapping strings to identifiers.--

REMARKS

The Specification

The present application for patent is a continuation of 08/957,512 filed on October 24, 1997. The specification, figures, and abstract are copies of that application as originally filed.

A number of minor typographical errors were discovered upon review of the Specification. The above amendment to the Specification corrects those errors. No new matter has been added.

Correction to Drawings:

Enclosed is a replacement sheet for sheet 4 of the drawings with corrections in red. Elements 46, 47, 48, and 49 were previously numbered incorrectly. However, the discussion of these elements on Page 17 of the Specification makes it clear that the intended numbering should be as on the Replacement drawing sheet.

Also enclosed are replacement sheets for Figures 16 and 18 with corrections in red. In both instances the arrow leading into element 166 incorrectly originated with element 165 rather than element 164. Again, the proper flow through the algorithm is evident from the Specification.

The Claims

Claims 1-105 are canceled herein. New Claims 145-188 are added herein. To avoid confusion with the claims in the parent application, Applicant has numbered the claims herein beginning with claim number 145. Claims 145-188 are pending in the application.

Conclusion

Consideration and allowance of Claims 145-188 is respectfully requested.

The additional claim fees have been indicated on the amendment transmittal letter. If Applicant is in error as to these fees, the Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account 19-0597.

Respectfully submitted,



Pehr B. Jansson

Registration No. 35,759

Date: October 23, 2001

Enclosures:

1. Version with Markings to Show Changes Made (3 pages)
2. Submission of Proposed Drawing Amendment (1 page)
3. Figs. 4, 16, & 18 with corrections marked in red ink (3 pages)

Customer No. 26751
Schlumberger Austin Technology Center
Attn: Pehr B. Jansson, Intellectual Property Law Dept.
8311 North FM 620
Austin, TX 78726
Tel: 512-331-3748
Fax: 512-331-3060

VERSION WITH MARKINGS TO SHOW CHANGES MADE

In the Specification:

Paragraph beginning at line 18 of page 3 has been amended as follows:

Each kind of memory is suitable for different purposes. Although ROM is the least expensive, it is suitable only for data that is unchanging, such as operating system code. EEPROM is useful for storing data that must be retained when power is removed, but is extremely slow to write. RAM can be written and read at high speed, but is expensive and data in RAM is lost when power is removed. A microprocessor system typically has relatively little ROM and EEPROM, and has 1 to 128 megabytes of RAM, since it is not constrained by what will fit on a single integrated circuit device, and often has access to an external disk memory system that serves as a large writable, non-volatile storage area at a lower cost [that]than EEPROM. However, a microcontroller typically has a small RAM of 0.1 to 2.0 K, 2K to 8K of EEPROM, and 8K – 56K of ROM.

Paragraph beginning at line 3 of page 5 has been amended as follows:

Due to the concern for security, applications written for integrated circuit cards have unique properties. For example, each application typically is identified with a particular owner or identity. Because applications typically are written in a low-level programming language, such as assembly language, the applications are written for a particular type of microcontroller. Due to the nature of low level programming languages, unauthorized applications may access data on the integrated circuit card. Programs written for [a]an integrated circuit card are identified with a particular identity so that if two identities want to perform the same programming function there must be two copies of some portions of the application on the microcontroller of the integrated circuit card.

Paragraph beginning at line 25 of page 6 has been amended as follows:

Among the advantages of the invention are one or more of the following. New applications may be downloaded to a smart card without compromising the security of the smart card. These applications may be provided by different companies loaded at

different times using different terminals. Security is not [comprised]compromised since the applications are protected against unauthorized access of any application code or data by the security features provided by the Java virtual machine. Smart card applications can be created in high level languages such as Java and Eiffel, using powerful mainstream program development tools. New applications can be quickly prototyped and downloaded to a smart card in a matter of hours without resorting to soft masks. Embedded systems using microcontrollers can also gain many of these advantages for downloading new applications, high level program development, and rapid prototyping by making use of this invention.

Paragraph beginning at line 24 of page 15 has been amended as follows:

The terminal 14 can also interact with applications running in the integrated circuit card 10. In some cases, different terminals may be used for these purposes. For example, one kind of terminal may be used to prepare applications, different terminals could be used to download the applications, and yet other terminals could be used to run the various applications. Terminals can be automated teller machines [(ATM)s] (ATMs), point-of-sale terminals, door security systems, toll payment systems, access control systems, or any other system that communicates with an integrated circuit card or microcontroller.

Paragraph beginning at line 6 of page 18 has been amended as follows:

To avoid dynamic linking in the card, all the information that is distributed across several Java class [file]files 24a, 24b, and 24c that form the application 24, are coalesced into one card class file 27 by the process shown in the flowchart in Fig. 5. The first class file to be processed is selected 51a. The constant pool 42 is compacted 51b in the following manner. All objects, classes, fields, methods referenced in a Java class file 24a are identified by using strings in the constant pool 42 of the class file 24a. The card class file converter 26 compacts the constant pool 42 found in the Java class file 24a into an optimized version. This compaction is achieved by mapping all the strings found in the class file constant pool 42 into integers (the size of which is microcontroller architecture

dependent). These integers are also referred to as IDs. Each ID uniquely identifies a particular object, class, field or method in the application 20. Therefore, the card class file converter 26 replaces the strings in the Java class file constant pool 42 with its corresponding unique ID. Appendix B shows an example application HelloSmartCard.java, with a table below illustrating the IDs corresponding to the strings found in the constant pool of the class file for this application. The IDs used for this example are 16-bit unsigned integers.

In the Claims:

Claims 1-105 have been cancelled. New Claims 145-188 have been added herein.

To avoid confusion with the claims in the parent application, Applicant has numbered the New Claims beginning with claim number 145. Claims 145-188 are pending in the application.

**SUBMISSION OF PROPOSED DRAWING AMENDMENT FOR APPROVAL BY
EXAMINER (37 CFR 1.121(a)(3)(ii) or 37 CFR 1.121(b)(3)(ii))**

Docket No.
40.0010 C1

In Re Application Of: **Wilkinson et al.**

Serial No.	Filing Date	Examiner	Art Unit
To be assigned		Chavis, J.	2122

Invention: **USING A HIGH LEVEL PROGRAMMING LANGUAGE WITH A MICROCONTROLLER**

Address to:
Assistant Commissioner for Patents
Washington, D.C. 20231

Attached please find:

(check applicable items)

- ☐ a sketch in permanent ink
- ☒ a copy of the original drawing(s) with red ink

showing proposed changes to the drawing(s) in this application for which the approval of the examiner is requested.

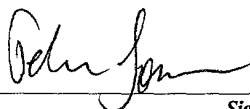
"EXPRESS MAIL" MAILING LABEL NO. EK857509565US

DATE OF DEPOSIT: October 23, 2001
I HEREBY CERTIFY UNDER 37 CFR 1.10 THAT THIS CORRESPONDENCE IS
BEING DEPOSITED WITH THE UNITED STATES POSTAL SERVICE AS "EXPRESS
MAIL POST OFFICE TO ADDRESSEE" WITH SUFFICIENT POSTAGE ON THE DATE
INDICATED ABOVE AND IS ADDRESSED TO THE ASSISTANT COMMISSIONER
FOR PATENTS WASHINGTON, D.C. 20231

SIGNATURE: Mary L. Thompson

TYPED OR PRINTED NAME OF PERSON SIGNING THIS CERTIFICATE

MARY L. Thompson



Signature

Customer No. 26751
Schlumberger Austin Technology Center
Attn: Pehr B. Jansson, Intellectual Prop Law Dept.
8311 North FM 620
Austin, TX 78726
Tel: 512-331-3748, Fax: 512-331-3060

Enclosure: Figs. 4, 16 & 18

Dated: October 23, 2001

I certify that this document and fee is being deposited
on _____ with the U.S. Postal Service as
first class mail under 37 C.F.R. 1.8 and is addressed to the
Assistant Commissioner for Patents, Washington, D.C.
20231.

Signature of Person Mailing Correspondence

Typed or Printed Name of Person Mailing Correspondence

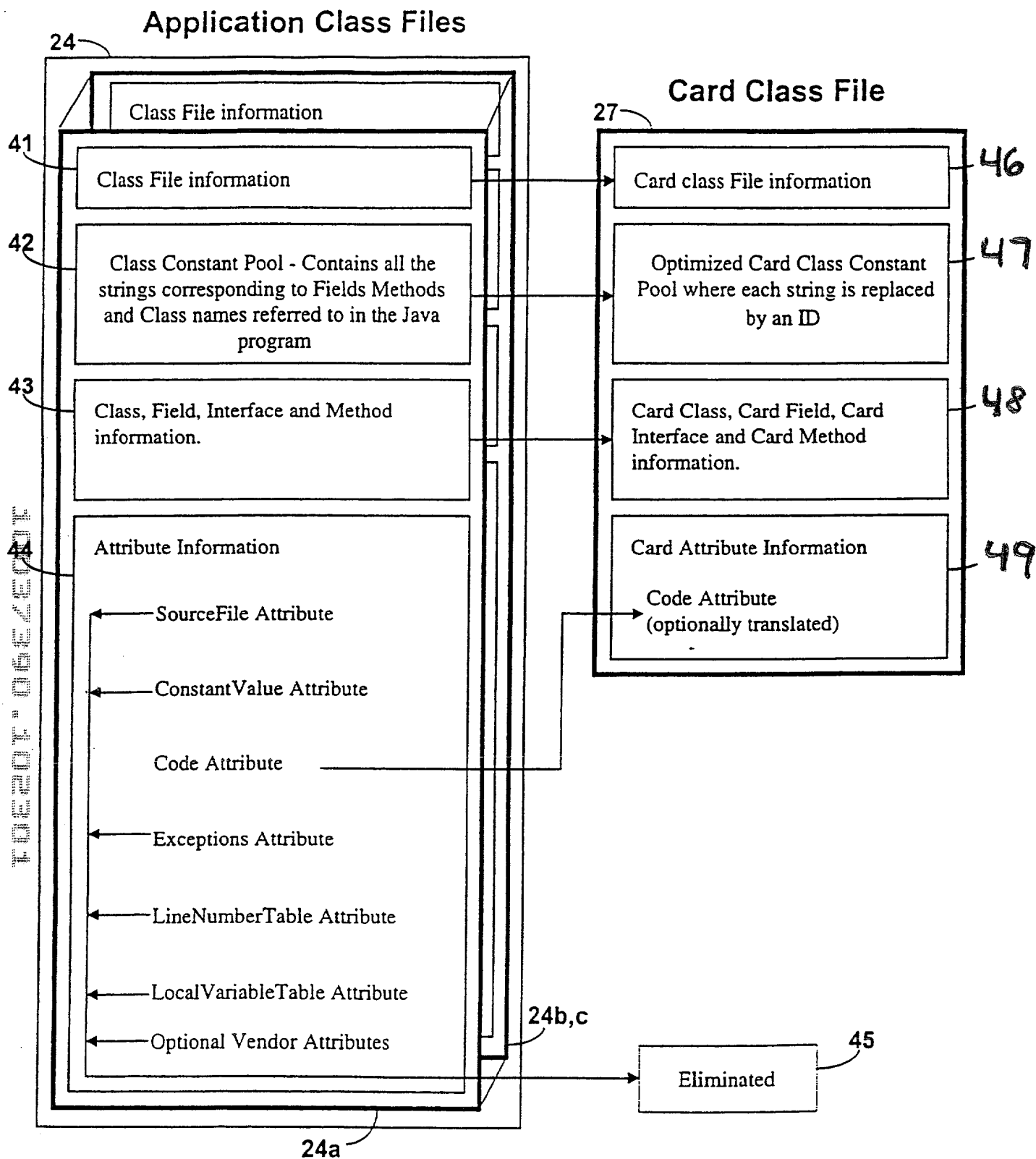
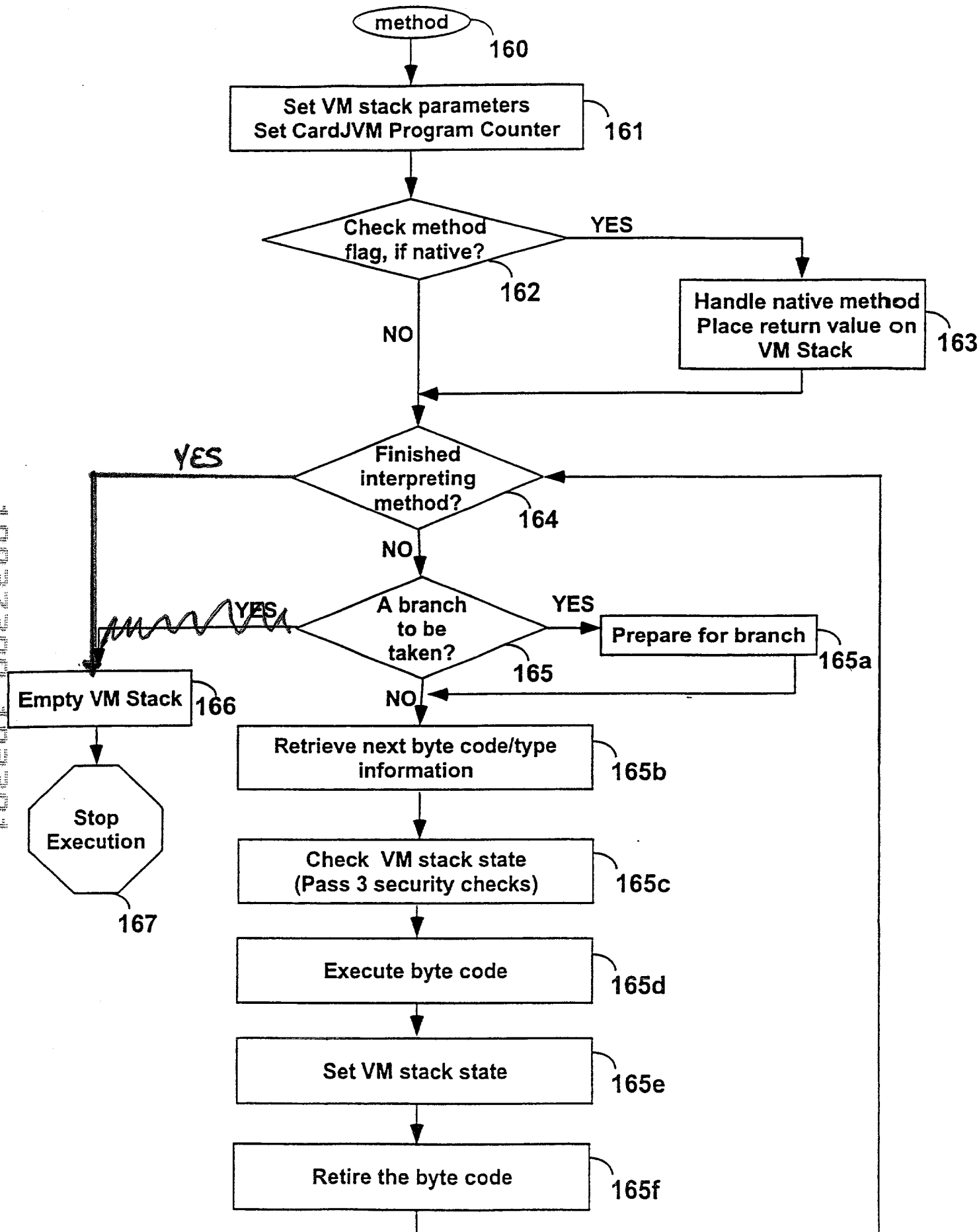


Fig. 4



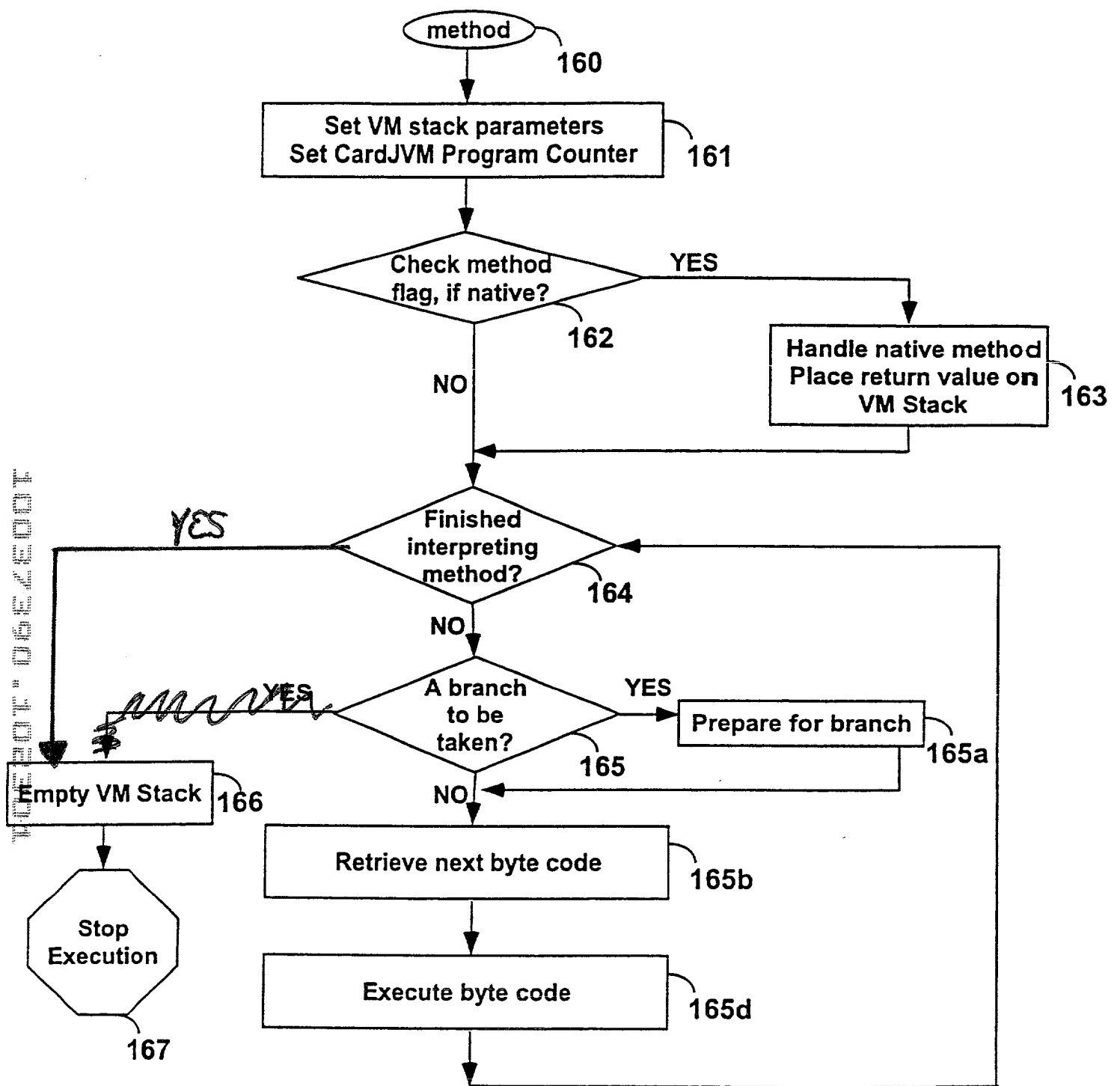


Fig. 18